Niharika Shetty

500754054

Section 9

1. Use Case Diagram

A use case diagram helps identify each sets of use cases and help determine the functions of the system. The application contains only two roles available: Customer and Manager. The manager controls two methods which include being able to add (Add Customer) and delete customers (Delete Customer) in the database. Whereas the customer has more functions which include depositing (Deposit Money) and withdrawing money (Withdraw Money), get the current balance (Get Balance) and make online purchases (Do Online Purchases). Both the manager and the customer can perform login (Login) and logout (Logout) functions.

Use Case Description

1. Name: Add Customer
2. Participating Actors: Manager
3. Entry Conditions: Manager is on the Add Customer page of the application
4. Exit Conditions: Manager has successfully added a new customer
5. Flow of Events:
    - Manager types a username in for the customer
    - Manager types a password in for the customer
    - System creates a new .txt file with the username and password information provided by the manager
    - New customer has successfully been added and can login as customer

2. Class Diagram

The class diagram is used to show the structure of the system. There are different classes, instance methods and instance variables of each type. The instance variables and methods can be defined with "+" with a public type and "-" with a private type. The ":" and what follows after is the instance type. There are also different relationships between each class. There are three types of relationships that can be used to describes the relationships between the classes. The "uses-a"

relationship is used to identify when a method of one class is using the object of another class. In this application, it can be seen that both the Manager class and Customer class uses the LoginController class. The "is-a" relationship is used to identify when one class extends the other. In this application, it can be seen that the Gold, Silver and Platinum class are extended by the Level class. The "has-a" relationship is used to identify a possessive hierarchy. In this application, it can be seen that the CustomerPageController class "has-a" Level class.

3. Class selected to add clauses: AddCustomerController

Javadocs created.

4. State Diagram

The state diagram used in this application executes the Customer class and further delegates Level to each Silver, Gold and Platinum.

Context **->** Customer

State -> Level

ConcreteState -> Silver, Gold, Platinum